PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-108656
(43)Date of publication of application : 12.04.2002

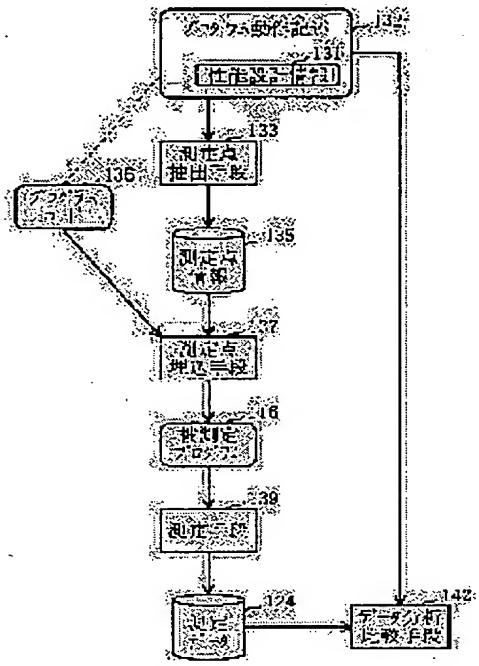
(51)Int.Cl. G06F 11/30

(21)Application number : 2000-298387
(22)Date of filing : 29.09.2000

(71)Applicant : NEC CORP
(72)Inventor : HORIKAWA TAKASHI

(54) PROGRAM OPERATION VERIFYING SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a program operation verifying system in which a measuring point can be arranged flexibly on a program to be measured and the operation situation of the program to be measured can be easily verified when it is executed on an actual device.
SOLUTION: A measuring point extracting means 133 extracts measuring points from program operation description 132 including performance design information 131 and a measuring point embedding means 137 embeds the measuring points (a start point and an end point) into a program code to prepare a program 116 to be measured. When this program is executed, a measuring means 139 outputs the time and event at each measuring point to generate measurement data 124. Thus, it is possible to learn the lapsed time between the measuring points when the program is executed. Therefore, it is possible to learn the operation situation and to verify the operation. This system can also be applied even to a performance index (for example, a CPU consuming time and the number of times of access of a disk or a network) other than the lapsed time.



LEGAL STATUS

[Date of request for examination] 21.08.2001
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2002－108656
（P2002－108656A）
(43)公開日　平成14年４月12日(2002.4.12)

(51)Int.Cl.'	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　11/30	３１０	Ｇ０６Ｆ　11/30	３１０Ａ　５Ｂ０４２

審査請求　有　　請求項の数９　ＯＬ　（全 14 頁）

(21)出願番号　特願2000－298387(P2000－298387)

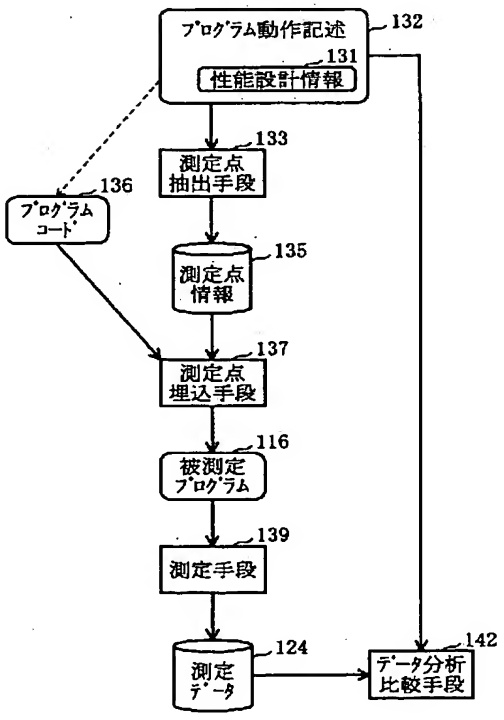
(22)出願日　平成12年９月29日(2000.9.29)

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(72)発明者　堀川　隆
東京都港区芝五丁目７番１号　日本電気株式会社内
(74)代理人　100083987
弁理士　山内　梅雄

Ｆターム(参考)　5B042 HH20 HH25 LA02 MA14 MA20
MB01 MB05 MB06 MC33

(54)【発明の名称】　プログラム動作検証システム

(57)【要約】　（修正有）
【課題】　被測定プログラム上に柔軟に測定点を配置することができ、被測定プログラムが実際の装置で実行されるときの動作状況を容易に検証することのできるプログラム動作検証システムを提供する。
【解決手段】　測定点抽出手段１３３は性能設計情報１３１を含んだプログラム動作記述１３２から測定点を抽出し、測定点埋込手段１３７によってプログラムコードに測定点（開始点および終了点）を埋め込んで被測定プログラム１１６を作成する。これを実行すると、測定手段１３９は各測定点でそれらの時刻とイベントを出力して測定データ１２４とする。測定データ１２４によってプログラム実行時の測定点間の経過時間が分かる。これにより、動作状況を知り動作検証を行うことができる。経過時間以外の性能指標（たとえばＣＰＵ消費時間、ディスクやネットワークのアクセス回数）にも本発明を適用できる。



1

【特許請求の範囲】

【請求項１】　プログラムのある特定範囲の処理に要する経過時間等の所定の性能指標を調べる性能設計情報を含んだプログラム動作記述から検証の対象となる被測定プログラムに埋め込む測定開始のための測定開始点および測定終了のための測定終了点を読み出す読出手段と、
この読出手段で読み出した測定開始点および測定終了点を該当する箇所にそれぞれ埋め込んだ被測定プログラムを作成する被測定プログラム作成手段と、
この被測定プログラム作成手段によって作成された被測定プログラムを所定の装置に組み込んで実行するプログラム実行手段と、
このプログラム実行手段の被測定プログラムの実行の際に前記測定開始点および測定終了点の埋め込み箇所の処理が実行されるときの性能指標をそれぞれ測定する性能指標測定手段と、
この時刻測定手段によって測定された測定開始点と測定終了点のそれぞれの性能指標の差から前記被測定プログラムのこれら測定開始点から測定終了点に至る性能指標値を算出する性能指標値算出手段と、
この性能指標値算出手段によって算出した性能指標値を前記被測定プログラムの対応する箇所と共に表示する表示手段とを具備することを特徴とするプログラム動作検証システム。

【請求項２】　プログラムのある特定範囲の処理に要する経過時間等の所定の性能指標がその処理について予め定めた設計値の範囲内に収まっているかどうかを少なくとも調べる性能設計情報を含んだプログラム動作記述から検証の対象となる被測定プログラムに埋め込む測定開始のための測定開始点と測定終了のための測定終了点および前記設計値を読み出す読出手段と、
この読出手段で読み出した測定開始点および測定終了点を該当する箇所にそれぞれ埋め込んだ被測定プログラムを作成する被測定プログラム作成手段と、
この被測定プログラム作成手段によって作成された被測定プログラムを所定の装置に組み込んで実行するプログラム実行手段と、
このプログラム実行手段の被測定プログラムの実行の際に前記測定開始点および測定終了点の埋め込み箇所の処理が実行されるときの性能指標をそれぞれ測定する性能指標測定手段と、
この性能指標測定手段によって測定された測定開始点と測定終了点のそれぞれの性能指標の差から前記被測定プログラムのこれら測定開始点から測定終了点に至る性能指標値を算出する性能指標値算出手段と、
この性能指標値算出手段によって算出した性能指標値を前記被測定プログラムの対応する箇所ならびに前記設計値と共に表示する表示手段とを具備することを特徴とするプログラム動作検証システム。

【請求項３】　プログラムのある特定範囲の処理に要する経過時間等の所定の性能指標がその処理について予め定めた設計値の範囲内に収まっているかどうかを少なくとも調べる性能設計情報を含んだプログラム動作記述から検証の対象となる被測定プログラムに埋め込む測定開始のための測定開始点と測定終了のための測定終了点および前記設計値を読み出す読出手段と、
この読出手段で読み出した測定開始点および測定終了点を該当する箇所にそれぞれ埋め込んだ被測定プログラムを作成する被測定プログラム作成手段と、
この被測定プログラム作成手段によって作成された被測定プログラムを所定の装置に組み込んで実行するプログラム実行手段と、
このプログラム実行手段の被測定プログラムの実行の際に前記測定開始点および測定終了点の埋め込み箇所の処理が実行されるときの性能指標をそれぞれ測定する性能指標測定手段と、
この性能指標測定手段によって測定された測定開始点と測定終了点の性能指標の差から前記被測定プログラムのこれら測定開始点から測定終了点に至るまでのプログラムの実行に要する性能指標値を算出する性能指標値算出手段と、
この性能指標値算出手段によって算出した性能指標値が前記設計値を満足するか否かを判別する判別手段と、
この判別手段によって判別した結果を前記被測定プログラムの対応する箇所と共に表示する表示手段とを具備することを特徴とするプログラム動作検証システム。

【請求項４】　性能指標は、プログラムのある特定範囲の処理に要する経過時間の他に、ＣＰＵ消費時間、ディスクやネットワークのアクセス回数、入出力データのサイズ等の指標であることを特徴とする請求項１～請求項３記載のプログラム動作検証システム。

【請求項５】　プログラムのある特定範囲の処理に要する経過時間がその処理について予め定めた設計値の範囲内に収まっているかどうかを少なくとも調べる性能設計情報を含んだプログラム動作記述から検証の対象となる被測定プログラムに埋め込む測定開始のための測定開始点と測定終了のための測定終了点および前記設計値を読み出す読出手段と、
この読出手段で読み出した測定開始点および測定終了点を該当する箇所にそれぞれ埋め込んだ被測定プログラムを作成する被測定プログラム作成手段と、
この被測定プログラム作成手段によって作成された被測定プログラムを所定の装置に組み込んで実行するプログラム実行手段と、
このプログラム実行手段の被測定プログラムの実行の際に前記測定開始点および測定終了点の埋め込み箇所の処理が実行されるときの時刻をそれぞれ測定する時刻測定手段と、
この時刻測定手段によって測定された測定開始点と測定終了点の時刻の差から前記被測定プログラムのこれら測

定開始点から測定終了点に至るまでのプログラムの実行に要する経過時間を算出する経過時間算出手段と、
この経過時間算出手段によって算出した経過時間が前記設計値を満足するか否かを判別する判別手段と、
この判別手段によって判別した結果を前記被測定プログラムの対応する箇所と共に表示する表示手段とを具備することを特徴とするプログラム動作検証システム。

【請求項６】　前記表示手段は経過時間算出手段によって算出した実際の経過時間も前記被測定プログラムの対応する箇所と共に表示することを特徴とする請求項５記載のプログラム動作検証システム。

【請求項７】　前記所定の装置は複数の装置によって構成されており、前記プログラムのある特定範囲の処理とは被測定プログラムのうち特定の装置に割り当てられたコンポーネントの処理であることを特徴とする請求項１～請求項３または請求項５記載のプログラム動作検証システム。

【請求項８】　前記所定の装置は通信ケーブルによって接続された複数の装置によって構成されており、それぞれの装置に対して前記被測定プログラムのコンポーネントが割り振られており、前記表示手段はそれぞれのコンポーネントについての表示を一箇所で集中的に行うことを特徴とする請求項１～請求項３または請求項５記載のプログラム動作検証システム。

【請求項９】　前記所定の装置は通信ケーブルによって接続された複数の装置によって構成されており、それぞれの装置に対して前記被測定プログラムのコンポーネントが割り振られており、前記表示手段はそれぞれの装置単位で該当するコンポーネントごとの表示を行うことを特徴とする請求項１～請求項３または請求項５記載のプログラム動作検証システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明はプログラムの動作を検証するためのプログラム動作検証システムに係わり、特にプログラムの開発時に実際の計算機等でその動作を検証するためのプログラム動作検証システムに関する。

【０００２】

【従来の技術】ソフトウェア工学の進展によって、比較的大きなソフトウェアをその構成部品としてのコンポーネントを組み合わせて開発するといったコンポーネント指向開発が普及しつつある。コンポーネントとは、ある機能やサービスを提供するための独立して交換可能なソフトウェアの構成単位をいう。

【０００３】このような開発手法をサポートする技術の１つにUML（Unified Modeling Language）がある。UMLはソフトウェアの仕様や、図示、体系化、文書化の方法を規定したシステム設計用のモデリング言語である。UMLを利用すると、高品質なシステム構築を可能にするだけでなく、リアルタイムシステムや分散システム等のさまざまなシステム設計を簡素化することができる。UMLはOMG（Object Management Group, Inc.）が標準化を行っている。このUMLで規定されている記述方法の１つであるシーケンス図は、作成したソフトウェアがコンピュータ等の装置で実際に処理されるときの動作を、前記したコンポーネントあるいはそれを構成するオブジェクトの間で行われる通信といった相互作用を時間軸に沿って記述するものである。

【０００４】このシーケンス図は、ソフトウェアの動作を時間軸で表わしているので、この図の中に時間に関係する性能情報を含めることが容易にできる。たとえばあるコンポーネントの処理は３秒間を要したといった情報を含めてソフトウェアの動作を表現することができる。実際に、UMLの第１．３版の仕様書（OMG UnifiedModeling Language Specification, http://cgi.omg.org/cgi-bin/doc?formal/00-03-0.1.pdf）の３－９６ページに示されているシーケンス図には、プログラムの実行に関する性能条件も一緒に記述している。このような性能条件の集まりを本明細書では性能設計情報と呼ぶことにする。

【０００５】一方、計算機の従来の性能測定は、マイクロソフト社のＯＳ（オペレーションシステム）としてのウィンドウズで使用されているパフォーマンスモニタ（performance monitor）やユニックス（ＵＮＩＸ（登録商標））におけるＳＡＲ（システム動作情報表示機能）のようにＣＰＵ（中央処理装置）やディスクといった計算機資源の使用率の計測が一般的である。

【０００６】これら従来から使用されたツールは、一定の周期で測定対象のシステムの状態を調べて計算機資源の使用率を見積もるサンプリング手法をベースとしている。また、計算機資源の使用開始や終了といったイベント発生時に測定操作を行うことで、計算機資源の使用時間を測定するようにしたイベントドリブン方式のツールも存在している。

【０００７】これらはいずれも、測定期間中の計算機資源の使用率（あるいは使用時間）を測定するものである。すなわちこれらの性能測定は、性能設計情報としてのシーケンス図で記述された性能条件に直接対応するものではない。したがって、たとえばあるオブジェクトが処理要求メッセージを受信してからリプライを返すまでの処理に要した計算機資源量を測定するといった、性能条件で引用されているプログラム実行途中の２点の処理を測定することはできない。このため、前記したようにソフトウェアを作成するときにその構成部品としてのコンポーネントごとの性能を評価するといったことができない。

【０００８】ところで、特開平１１－９６０４６号公報ではプログラムが動作する計算機システムからプログラムの稼動状況を取得する際に、予め該当するプログラム、収集タイミング、収集するデータの項目等を含んだ

収集情報選択定義情報を外部から定義情報格納手段に格納するようにしている。この技術では、熟練したソフトウェア開発者が収集情報選択定義情報を定義情報格納手段に格納することでこれらの者のノウハウを活かして誤った情報を収集したり不十分な情報を収集するといった不都合を解消している。しかしながら、このような技術で活用される定義情報格納手段では該当するプログラムを指定するのは人手に頼ることになる。また、熟練者のノウハウを活用するということは定義情報格納手段に格納する情報の入力を行う者は熟練者であることを必要とするという問題がある。

【０００９】一方、特開平９－１８３０３号公報では、現在使用されているプログラム言語よりも上位レベルの記述を利用して、分散アプリケーションを自動構築する手法の分野で、上位レベル記述から測定点を抽出して、プローブを自動的に埋め込む処理を開示している。また、特許第２９２４４３６号公報では、性能測定対象のプロセスに遠隔手続き呼び出しがあることを前もって検出して性能の測定を行うシステムを開示している。

【００１０】更に特開平８－３１４７６７号公報では、ＯＳ機能として用意されているイベント収集機能を使用して計算機の性能を測定する技術を開示している。すなわち、この技術ではタスクの実行中にＯＳによって収集されるイベント情報を時系列的に格納して、全タスクの実行後に、これらのイベント情報を順次読み出して、タスクごとにデータベースのリソースや入力操作に関するデータベース関連のイベント情報を抽出し、データベースの操作性に関する複数の評価パラメータを集計するようにしている。

【００１１】

【発明が解決しようとする課題】しかしながら、これらの技術は先の特開平１１－９６０４６号公報とも共通するが、プログラムにおける測定点の抽出をソフトウェア技術者あるいは測定実施者が手作業で行っていたり、メッセージの送受信点といったように予め決まったルールを使用したり、ＯＳ自体が用意している機能を利用するようになっている。

【００１２】このため、手作業で行う場合には得られた測定結果に人為的な影響が作用してしまい、測定結果が妥当であるかどうかの判断が困難であるという問題があった。また、測定の実施者が特別な知識や経験を有する者に限定されるだけでなく、作業に時間を要するといった問題があった。更に、予め決まったルールや機能を使用する場合には、手作業で行う場合の問題がなくなる一方で、プログラム上に測定点を設置することができる位置が、これらのルールや機能を適用できる箇所に限られることになる。したがって、測定を柔軟に行うことができないという問題があった。

【００１３】このため、性能設計情報を含むプログラム動作記述をもとに開発されたプログラムがこれらの性能

設計情報で示されている性能条件（たとえば、プログラム実行中における任意の点を参照する）を満足しているかどうかを調べるための測定を行うものとすると、従来ではソフトウェアの開発者が手作業で性能設計情報に対応する測定点を抽出していた。そして、これらの測定点を用いて、プログラムの実行に要した時間が適切な時間範囲であったかといった性能測定を実施する必要があった。このために、前記した手作業が原因で測定の実施に手間がかかるといった問題があった。

【００１４】またこのような従来の手法によると、測定点の抽出操作を個々のプログラムごとに個別に行う必要があった。このように、性能設計情報に対応させた測定を行う測定手法を汎用化された測定ツールという形で提供するまでに至っておらず、これが測定の実施に時間がかかる一因ともなっていた。また、予め決まったルールや機能を使用する場合には、測定点が一方的に定められることになり、必要な測定点を任意に抽出することができない。したがって、性能設計を行う際に性能条件を自由に記述することができないという問題があった。

【００１５】そこで本発明の目的は、被測定プログラム上に柔軟に測定点を配置することができ、被測定プログラムが実際の装置で実行されるときの動作状況を容易に検証することのできるプログラム動作検証システムを提供することにある。

【００１６】

【課題を解決するための手段】請求項１記載の発明では、（イ）プログラムのある特定範囲の処理に要する経過時間等の所定の性能指標を調べる性能設計情報を含んだプログラム動作記述から検証の対象となる被測定プログラムに埋め込む測定開始のための測定開始点および測定終了のための測定終了点を読み出す読出手段と、

（ロ）この読出手段で読み出した測定開始点および測定終了点を該当する箇所にそれぞれ埋め込んだ被測定プログラムを作成する被測定プログラム作成手段と、（ハ）この被測定プログラム作成手段によって作成された被測定プログラムを所定の装置に組み込んで実行するプログラム実行手段と、（ニ）このプログラム実行手段の被測定プログラムの実行の際に測定開始点および測定終了点の埋め込み箇所の処理が実行されるときの性能指標をそれぞれ測定する性能指標測定手段と、（ホ）この時刻測定手段によって測定された測定開始点と測定終了点のそれぞれの性能指標の差から被測定プログラムのこれら測定開始点から測定終了点に至る性能指標値を算出する性能指標値算出手段と、（ヘ）この性能指標値算出手段によって算出した性能指標値を被測定プログラムの対応する箇所と共に表示する表示手段とをプログラム動作検証システムに具備させる。

【００１７】すなわち請求項１記載の発明では、読出手段によって性能設計情報を含んだプログラム動作記述から検証の対象となる被測定プログラムに埋め込む測定開

始点と測定終了点および設計値を読み出して、被測定プログラム作成手段で測定開始点と測定終了点をプログラムの該当箇所に埋め込んで被測定プログラムを作成するようにしている。そして、時刻測定手段によってプログラム実行手段の被測定プログラムの実行の際に測定開始点および測定終了点の埋め込み箇所の処理が実行されるときの性能指標をそれぞれ測定し、それぞれの性能指標の差から被測定プログラムのこれら測定開始点から測定終了点に至る性能指標値を算出するようにしている。算出した性能指標値は判別手段を用いて設計値を満足するか否かを判別し、判別結果が被測定プログラムの対応する箇所に表示される。このように請求項１記載の発明では、プログラム動作記述の性能設計情報に記述した内容を基にして測定開始点と測定終了点を抽出するので、たとえばプログラマが性能設計情報として測定の開始点や終了点を判読できるデータを書き込んでおくことにより、被測定プログラムの所望の箇所で測定を開始したり終了させることができる。したがって、プログラムにおける測定点の抽出をソフトウェア技術者あるいは測定実施者が手作業で行う必要がない。また、メッセージの送受信点といったように予め決まったルールを使用したり、ＯＳ自体が用意している機能を利用する場合と異なり、被測定プログラムの所望のコンポーネントの開始点と終了点を任意に指定して測定を行うことができる。そしてこれらの点での性能指標を求め、これらの差から性能指標値を算出しこれを設計値と比較した判別結果を被測定プログラムの対応する箇所に対応付けて表示するので、チェックするコンポーネントが多いような場合にもそれぞれの性能指標値が設計値を満足しているかどうかを短時間にチェックすることができる。

【００１８】請求項２記載の発明では、（イ）プログラムのある特定範囲の処理に要する経過時間等の所定の性能指標がその処理について予め定めた設計値の範囲内に収まっているかどうかを少なくとも調べる性能設計情報を含んだプログラム動作記述から検証の対象となる被測定プログラムに埋め込む測定開始のための測定開始点と測定終了のための測定終了点および設計値を読み出す読出手段と、（ロ）この読出手段で読み出した測定開始点および測定終了点を該当する箇所にそれぞれ埋め込んだ被測定プログラムを作成する被測定プログラム作成手段と、（ハ）この被測定プログラム作成手段によって作成された被測定プログラムを所定の装置に組み込んで実行するプログラム実行手段と、（ニ）このプログラム実行手段の被測定プログラムの実行の際に測定開始点および測定終了点の埋め込み箇所の処理が実行されるときの性能指標をそれぞれ測定する性能指標測定手段と、（ホ）この性能指標測定手段によって測定された測定開始点と測定終了点のそれぞれの性能指標の差から被測定プログラムのこれら測定開始点から測定終了点に至る性能指標値を算出する性能指標値算出手段と、（ヘ）この性能指標値算出手段によって算出した性能指標値を被測定プログラムの対応する箇所ならびに設計値と共に表示する表示手段とをプログラム動作検証システムに具備させる。

【００１９】すなわち、請求項２記載の発明では、プログラム動作記述の性能設計情報に記述した内容を基にして測定開始点と測定終了点を抽出するので、たとえばプログラマが性能設計情報として測定の開始点や終了点を判読できるデータを書き込んでおくことにより、被測定プログラムの所望の箇所で測定を開始したり終了させることができる。したがって、プログラムにおける測定点の抽出をソフトウェア技術者あるいは測定実施者が手作業で行う必要がない。また、メッセージの送受信点といったように予め決まったルールを使用したり、ＯＳ自体が用意している機能を利用する場合と異なり、被測定プログラムの所望のコンポーネントの開始点と終了点を任意に指定して測定を行うことができる。そしてこれらの点での性能指標を求め、これらの差から性能指標値を算出して表示するので、性能指標値を被測定プログラムの対応する箇所に対応付けて簡単にチェックすることができる。更に本発明では、算出した性能指標値を設計値と共に被測定プログラムの対応する箇所に表示するので、算出した性能指標値が設計値を満足しているか否かを簡単にチェックすることができる。

【００２０】請求項３記載の発明では、（イ）プログラムのある特定範囲の処理に要する経過時間等の所定の性能指標がその処理について予め定めた設計値の範囲内に収まっているかどうかを少なくとも調べる性能設計情報を含んだプログラム動作記述から検証の対象となる被測定プログラムに埋め込む測定開始のための測定開始点と測定終了のための測定終了点および設計値を読み出す読出手段と、（ロ）この読出手段で読み出した測定開始点および測定終了点を該当する箇所にそれぞれ埋め込んだ被測定プログラムを作成する被測定プログラム作成手段と、（ハ）この被測定プログラム作成手段によって作成された被測定プログラムを所定の装置に組み込んで実行するプログラム実行手段と、（ニ）このプログラム実行手段の被測定プログラムの実行の際に測定開始点および測定終了点の埋め込み箇所の処理が実行されるときの性能指標をそれぞれ測定する性能指標測定手段と、（ホ）この性能指標測定手段によって測定された測定開始点と測定終了点の性能指標の差から被測定プログラムのこれら測定開始点から測定終了点に至るまでのプログラムの実行に要する性能指標値を算出する性能指標値算出手段と、（ヘ）この性能指標値算出手段によって算出した性能指標値が設計値を満足するか否かを判別する判別手段と、（ト）この判別手段によって判別した結果を被測定プログラムの対応する箇所と共に表示する表示手段とをプログラム動作検証システムに具備させる。

【００２１】すなわち請求項３記載の発明では、プログラム動作記述の性能設計情報に記述した内容を基にして

測定開始点と測定終了点を抽出するので、たとえばプログラマが性能設計情報として測定の開始点や終了点を判読できるデータを書き込んでおくことにより、被測定プログラムの所望の箇所で測定を開始したり終了させることができる。したがって、プログラムにおける測定点の抽出をソフトウェア技術者あるいは測定実施者が手作業で行う必要がない。また、メッセージの送受信点といったように予め決まったルールを使用したり、ＯＳ自体が用意している機能を利用する場合と異なり、被測定プログラムの所望のコンポーネントの開始点と終了点を任意に指定して測定を行うことができる。そしてこれらの点での性能指標を求め、これらの差から性能指標値を算出しこれを設計値と比較した判別結果を被測定プログラムの対応する箇所に対応付けて表示するので、チェックするコンポーネントが多いような場合にもそれぞれの性能指標値が設計値を満足しているかどうかを短時間にチェックすることができる。

【００２２】請求項４記載の発明では、性能指標は、たとえばプログラムのある特定範囲の処理に要する経過時間の他に、ＣＰＵ消費時間、ディスクやネットワークのアクセス回数、入出力データのサイズ等の指標であることを示している。

【００２３】請求項５記載の発明では、（イ）プログラムのある特定範囲の処理に要する経過時間がその処理について予め定めた設計値の範囲内に収まっているかどうかを少なくとも調べる性能設計情報を含んだプログラム動作記述から検証の対象となる被測定プログラムに埋め込む測定開始のための測定開始点と測定終了のための測定終了点および設計値を読み出す読出手段と、（ロ）この読出手段で読み出した測定開始点および測定終了点を該当する箇所にそれぞれ埋め込んだ被測定プログラムを作成する被測定プログラム作成手段と、（ハ）この被測定プログラム作成手段によって作成された被測定プログラムを所定の装置に組み込んで実行するプログラム実行手段と、（ニ）このプログラム実行手段の被測定プログラムの実行の際に測定開始点および測定終了点の埋め込み箇所の処理が実行されるときの時刻をそれぞれ測定する時刻測定手段と、（ホ）この時刻測定手段によって測定された測定開始点と測定終了点の時刻の差から被測定プログラムのこれら測定開始点から測定終了点に至るまでのプログラムの実行に要する経過時間を算出する経過時間算出手段と、（ヘ）この経過時間算出手段によって算出した経過時間が設計値を満足するか否かを判別する判別手段と、（ト）この判別手段によって判別した結果を被測定プログラムの対応する箇所と共に表示する表示手段とをプログラム動作検証システムに具備させる。

【００２４】すなわち請求項５記載の発明では、読出手段によって性能設計情報を含んだプログラム動作記述から検証の対象となる被測定プログラムに埋め込む測定開始点と測定終了点および設計値を読み出して、被測定プログラム作成手段で測定開始点と測定終了点をプログラムの該当箇所に埋め込んで被測定プログラムを作成するようにしている。そして、時刻測定手段によってプログラム実行手段の被測定プログラムの実行の際に測定開始点および測定終了点の埋め込み箇所の処理が実行されるときの時刻をそれぞれ測定し、被測定プログラムのこれら測定開始点から測定終了点に至るまでのプログラムの実行に要する経過時間を経過時間算出手段によって算出するようにしている。算出した経過時間は判別手段を用いて設計値を満足するか否かを判別し、判別結果が被測定プログラムの対応する箇所に表示される。

【００２５】このように請求項５記載の発明では、プログラム動作記述の性能設計情報に記述した内容を基にして測定開始点と測定終了点を抽出するので、たとえばプログラマが性能設計情報として測定の開始点や終了点を判読できるデータを書き込んでおくことにより、被測定プログラムの所望の箇所で測定を開始したり終了させることができる。したがって、プログラムにおける測定点の抽出をソフトウェア技術者あるいは測定実施者が手作業で行う必要がない。また、メッセージの送受信点といったように予め決まったルールを使用したり、ＯＳ自体が用意している機能を利用する場合と異なり、被測定プログラムの所望のコンポーネントの開始点と終了点を任意に指定して測定を行うことができる。そしてこれらの点での処理時刻を求め、これらの差から経過時間を算出しこれを設計値と比較した判別結果を被測定プログラムの対応する箇所に対応付けて表示するので、チェックするコンポーネントが多いような場合にもそれぞれの所要時間が設計値を満足しているかどうかを短時間にチェックすることができる。

【００２６】請求項６記載の発明では、請求項５記載のプログラム動作検証システムで、表示手段は経過時間算出手段によって算出した実際の経過時間も被測定プログラムの対応する箇所と共に表示することを特徴としている。

【００２７】すなわち請求項６記載の発明では、請求項３記載の発明による表示内容に加えて経過時間算出手段によって算出した実際の経過時間も表示するようにしているので、この経過時間を設計値と対比することでプログラムの実行について何らかの不具合があったときにその原因を大まかに類推することができる。

【００２８】請求項７記載の発明では、請求項１～請求項３または請求項５記載のプログラム動作検証システムで、前記した所定の装置は複数の装置によって構成されており、プログラムのある特定範囲の処理とは被測定プログラムのうち特定の装置に割り当てられたコンポーネントの処理であることを特徴としている。

【００２９】すなわち請求項７記載の発明では、１つの被測定プログラムが複数の装置によってコンポーネント単位で実行される例を示している。このようなコンポー

ネントごとの経過時間を表示したりそれぞれの設計値と比較することがコンポーネント単位で測定開始点と測定終了点をプログラム動作記述から抽出し被測定プログラムに埋め込むことで実現可能になる。

【００３０】請求項８記載の発明では、請求項１～請求項３または請求項５記載のプログラム動作検証システムで、前記した所定の装置は通信ケーブルによって接続された複数の装置によって構成されており、それぞれの装置に対して被測定プログラムのコンポーネントが割り振られており、表示手段はそれぞれのコンポーネントについての表示を一箇所で集中的に行うことを特徴としている。

【００３１】すなわち請求項８記載の発明では、１つの被測定プログラムが複数の装置によって実行されるとき、それぞれの装置におけるコンポーネントごとの検証結果は１つの装置で集中的に表示される例を示しており、これにより被測定プログラムの動作全体についての集中的な検証が可能になる。

【００３２】請求項９記載の発明では、請求項１～請求項３または請求項５記載のプログラム動作検証システムで、前記した所定の装置は通信ケーブルによって接続された複数の装置によって構成されており、それぞれの装置に対して被測定プログラムのコンポーネントが割り振られており、表示手段はそれぞれの装置単位で該当するコンポーネントごとの表示を行うことを特徴としている。

【００３３】すなわち請求項９記載の発明では、１つの被測定プログラムが複数の装置によって実行されるとき、それぞれの装置におけるコンポーネントごとの検証結果は装置ごとに表示される例を示している。これにより、各装置側で自己に関係するコンポーネントの不具合に対するハードウェア側の原因を探る等の対処を容易に行うことができる。

【００３４】

【発明の実施の形態】

【００３５】

【実施例】以下実施例につき本発明を詳細に説明する。

【００３６】図１は本発明の一実施例におけるプログラム動作検証システムの構成の概要を示したものである。本実施例のシステムは、被測定計算機１０１と測定装置１０２とによって構成されている。被測定計算機１０１は、通常のパーソナルコンピュータと同一の構成となっており、ＣＰＵ（中央処理装置）１１１およびハードディスク、ＲＡＭ（ランダム・アクセス・メモリ）等からなるメモリ１１２の他に、測定装置１０２にデータを出力するためのデータ出力部１１３を備えており、それらがバス１１４に接続されている。メモリ１１２内にはこの被測定計算機１０１の基本的な制御を行わせるためのプログラムとしてのＯＳ１１５とこのＯＳ１１５上で実行される被測定プログラム１１６が格納されている。被測定プログラム１１６には、プログラム動作記述を基にして測定点が埋め込まれている。

【００３７】なお、この図ではキーボード等の入力機器（図示せず）やディスク読取装置等の装置からデータの入力を行うデータ入力部やディスプレイ（図示せず）の表示制御を行う表示制御部等の回路装置は本発明の動作と直接関係しない。そこで、これらの図示を省略している。

【００３８】被測定計算機１０１のＯＳ１１５の基本機能を実現する部分としてのカーネル内には、ＣＰＵや磁気ディスク、光ディスクといった計算機資源の使用状況を示すイベントを出力するためにカーネルプローブ１１７が埋め込まれている。被測定プログラム１１６が実行され、計算機資源の使用状況のうちの特定のものがカーネルプローブ１１７によって検出されると、これらの使用状況がイベントデータ１１８として測定装置１０２に出力される。このカーネルプローブ１１７の他にプログラム中に埋め込まれた測定点を検出するプローブ（以下、測定点プローブという。）も存在している。測定点プローブもこれが実行されたときイベントデータ１１８をデータ出力線１１９を介して測定装置１０２に出力するようになっている。

【００３９】一方、測定装置１０２はデータ出力線１１９によって受信した被測定計算機からのイベントデータ１１８を記録装置１２１に入力するようになっている。記録装置１２１には測定装置１０２内に設けられた時計回路１２２から時刻データ１２３も供給されるようになっている。記録装置１２１はイベントデータ１１８と時刻データ１２３を対応付けて測定データ１２４として記録する。この測定データ１２４を基にして、ある測定点から他の測定点に至るまでの経過時間が検証したい動作条件としての時間範囲（設計値）に合致しているかどうかの判別が行われる。これについては次に説明する。

【００４０】図２は、本実施例のプログラム動作検証システムの動作の概要を示したものである。図１に示した被測定プログラム１１６を作成するために、プログラムの作成時にプログラマは性能設計情報１３１を含んだプログラム動作記述１３２を作成する。プログラム動作記述１３２には、予めプログラマがプログラムの動作を実際の装置で検証するための測定の開始点および終了点を示す位置を記している。プログラムの実行に関する性能条件を表わした性能設計情報１３１として、本実施例では測定点による測定の開始から終了までに許容できる所要時間としての設計値が記述されている。

【００４１】測定点抽出手段１３３は、各コンポーネントの境界位置のような測定点をプログラム動作記述１３２から抽出し、これを所定の記憶領域に測定点情報１３５として格納する。これらの測定点情報１３５は、プログラムコード１３６の該当する位置に測定点埋込手段１３７によって埋め込まれる。被測定計算機１０１内で測

定点埋込手段１３７はＯＳ１１５によって所定の制御を行うＣＰＵ１１１が該当する。このような埋め込み操作によって被測定プログラム１１６（図１参照）が作成される。

【００４２】被測定プログラム１１６は測定点から他の測定点まで所要時間を測定する装置（たとえば図１に示した測定装置１０２）からなる測定手段１３９によって測定され、その結果としての測定データ１２４が作成される。測定データ１２４は、データ分析比較手段１４２に入力される。図１ではこれを記録装置１２１が兼用している。データ分析比較手段１４２は、プログラム動作記述１３２内の性能設計情報１３１から測定点の開始から終了までに予想される時間の許容値（設計値）と実際に費やした時間を比較して、被測定プログラム１１６の動作が正常であるかどうかの検証結果を図１に示す記録装置１２１のような装置でプリントアウトしたり、図示しないディスプレイに表示するようになっている。

【００４３】図３は、本実施例のプログラム検証用のクライアントとサーバの関係を示したものである。クライアント１６１はインターネットで通信を行うための通信ケーブル１６２を介してサーバ１６３と接続されている。なお、ここでは説明を簡単にするためにクライアント１６１とサーバ１６３を１対１の関係で図示している。図１に示した被測定計算機１０１と測定装置１０２は、図示していないが通信ケーブルでこれらクライアント１６１およびサーバ１６３と接続されている。

【００４４】図４は、プログラマによるプログラム動作記述の例を示したものである。本実施例ではＵＭＬ（Unified Modeling Language）を使用して記述を行っている。図で左側がクライアント１６１側の記述であり、右側がサーバ１６３側の記述である。これらの記述で、枠で示した“{”と“}”で囲まれた箇所が性能設計情報１７１～１７３である。

【００４５】この言語の記述を簡単に説明すると、“Begin Client Process”という記述によってクライアント１６１が処理を開始し、“Send Request To Server”という記述でたとえばある処理をサーバ１６３側に要求している。この“Send RequestTo Server”という処理の要求に付帯して、以下のような性能設計情報１７１が記述されている。

{Send Request To Server - Begin Client Process < 1 sec}　　（性能設計情報１７１）

【００４６】この性能設計情報１７１は、不等式の左辺で示した２点間、すなわち“BeginClient Process”から“Send Request To Server”までの間に消費するクライアント１６１側のＣＰＵの経過時間が右辺に示した“1 sec”（１秒）よりも短いという条件を示している。この条件は、プログラムを処理する装置側等に何らかの不具合がなければ、達成される範囲の値である前記した設計値として記載される。したがって、この場合には、“Begin Client Process”と“Send Request ToServer”の記述が行われたプログラムの箇所がそれぞれ測定点１８１、１８２となる。

【００４７】“Send Request To Server”という処理の要求の記述に基づいて、クライアント１６１側からサーバ１６３側にリクエスト（Request）が行われる。サーバ１６３では、“Receive Request From Client”という記述の箇所でクライアント１６１から処理の要求を受け、“Send Reply To Client”という記述の箇所で返答（Reply）をクライアント１６１に返す。サーバ１６３側におけるこの箇所には以下のような性能設計情報１７２が記述されている。

{Send Reply To Client - Receive Request From Client < 3 sec}　　（性能設計情報１７２）

【００４８】この性能設計情報１７２は、不等式の左辺で示した２点間、すなわち“Receive Request From Client”から“Send Reply To Client”までに消費するサーバ１６３側の経過時間が、右辺に示した“3 sec”（３秒）よりも短いという条件を示している。この条件も、プログラムを処理する装置側等に何らかの不具合がなければ、達成される範囲の値である前記した設計値として記載される。したがって、この場合には、“Receive Request From Client”と“Send Reply To Client”の記述が行われたプログラムの箇所がそれぞれ測定点１８３、１８４となる。

【００４９】クライアント１６１は“Receive Reply From Server”という記述の箇所でサーバ１６３側の返答（Reply）を受ける。そしてそれから所定の作業を行った後に“End Client Process”で所定の処理を完了させる。この記述の箇所には以下のような性能設計情報１７３が記述されている。

{End Client Process - Receive Reply From Server < 2 sec}　　（性能設計情報１７３）

【００５０】この性能設計情報１７３は、不等式の左辺で示した２点間、すなわち“End Client Process”から“Receive Reply From Server”までに消費するクライアント１６１の経過時間が右辺に示した“2sec”（２秒）よりも短いという条件を示している。この条件も、プログラムを処理する装置側等に何らかの不具合がなければ、達成される範囲の値である前記した設計値として記載される。したがって、この場合には、“Receive Reply From Server”と“End Client Process”の記述が行われたプログラムの箇所がそれぞれ測定点１８５、１８６となる。

【００５１】図１に示したＣＰＵ１１１はＯＳ１１５によって、このようなＵＭＬによる記述を解読する。そして、クライアント１６１側であれば、対応するそれぞれの測定点１８１、１８２、１８５、１８６と性能設計情報１７１、１７３をそのメモリ１１２に一時的に格納し、これを基にしてプログラム動作の検証を行うことに

なる。サーバ１６３側も同様の制御を行うがその結果を独自にサーバ１６３側で表示することもできるし、クライアント１６１側に返すような処理形態を採ることも可能である。

【００５２】図５は、クライアント側のプログラムコードとしてのクライアントプログラムの流れの要部を示したものである。まず、図４の測定点１８１に対応するプリプロセス（Pre Process）（ステップＳ２０１）から測定点１８２に対応するサーバ１６３側への処理要求（Send Request To Server）（ステップＳ２０２）へ移行して、更にサーバ１６３側からの返答（Receive Reply From Server）（ステップＳ２０３）を受信し、所定のプロセスを終了させる（Post Process）（ステップＳ２０４）ことになる。

【００５３】図６は、サーバ側のプログラムコードとしてのサーバプログラムの流れの要部を示したものである。サーバは、クライアント１６１から処理の要求を受ける（Receive Request From Client）（ステップＳ２２１）と、その要求された処理を実行し（Process Request）（ステップＳ２２２）、クライアント１６１に返答する（Send Reply To Client）（ステップＳ２２３）という処理を繰り返す。したがって、サーバ１６３のプログラムには実行の終了点は存在しない。

【００５４】次に図２に示した測定点抽出手段１３３による測定点の抽出動作を具体的に説明する。図１に示したＣＰＵ１１１とＯＳ１１５によって実現する測定点抽出手段１３３は、プログラム動作記述内の性能設計情報１７１～１７３を用いてそれぞれの測定点１８１～１８６を抽出するようになっている。図４に示した例の場合には、性能設計情報１７１～１７３として記述されている不等式の左辺で引用されている点、すなわち“Begin Client Process”、“Send Request To Server”、“Receive Request From Client”、“Send Reply To Client”、“ReceiveReply From Server”および“End Client Process”の６点が測定点１８１～１８６を表わした測定点情報として抽出される。

【００５５】図７は、これら抽出された測定点情報を埋め込んで作成した被測定プログラムを用いたクライアント側の処理を示したものであり、図８はイベントを出力するそれぞれのプローブとソフトウェアイベントを対応させて示したものである。図７の処理は図５の処理と対応している。クライアント１６１は、前記したプログラム中に埋め込まれた測定点を検出する測定点プローブの１つとしてのプローブ“Prb Cb”によってプリプロセス（Pre Process）を検出し（ステップＳ２４１）、クライアント処理を開始する。このとき、プローブ“Prb Cb”によってクライアント処理が開始した旨のイベントデータがデータ出力部１１３（図１）を介して被測定計算機１０１から測定装置１０２に送られる。測定装置１０２では、このイベントデータおよび時計回路１２２の出力する時刻データ１２３を記録装置１２１に送り、イベントデータごとに時刻を記録して測定データ１２４を作成していく。

【００５６】さて、図７のステップＳ２４１で測定点のうちの開始点が検出されたら、次の測定点プローブ“Prb Cs”によってサーバ１６３への処理要求（Send Request To Server）が検出されて（ステップＳ２４２）、その旨のイベントデータがデータ出力部１１３から測定装置１０２に送られる。この後、ステップＳ２４３でサーバ１６３側から返答を受ける（Receive Reply From Server）まで、クライアント１６１はスリープ（Sleep）状態となる。これはカーネルプローブ１１７で検出されて、同様にそのイベントが記録装置１２１側に送られて記録される。また、サーバ１６３側はクライアント１６１からの処理要求（Send Request To Server）によってウエイクアップ（Wake Up）するが、同様にこれはカーネルプローブ１１７で検出されて、記録装置１２１側に送られて記録されることになる。

【００５７】ステップＳ２４３でサーバ１６３側からの返答（Receive Reply From Server）があると、測定点プローブの１つとしてのプローブ“Prb Cr”がこれを検出して、同様に図８に示すようなイベントデータを出力する。そしてその後、サーバ１６３の返答に基づいた所定のプロセスを終了させる（Post Process）（ステップＳ２０４）と、測定点プローブの１つとしてのプローブ“Prb Ce”がこれを検出して、同様に図８に示すようなイベントデータを出力することになる。

【００５８】図９は、被測定プログラムを用いたサーバ側の処理を示したものであり、図６の処理と対応している。ステップＳ２６１ではクライアント１６１から要求を受信（Receive Request From Client）し、これが測定点プローブの１つとしてのプローブ“Prb Sr”で検出される。プローブ“Prb Sr”は図８に示したように、サーバ１６３がクライアント１６１から処理要求を受信した旨のイベントデータを出力する。ステップＳ２６２ではサーバ１６３が要求された処理を実行（Process Request）する。処理が終了すると、測定点プローブの１つとしてのプローブ“Prb Ss”がこれを検出して、クライアントに返答する（Send Reply To Client）（ステップＳ２６３）。この後、サーバがスリープ（Sleep）状態となり、クライアント１６１側がウエイクアップ（Wake Up）する。これらはカーネルプローブ１１７で検出されて、記録装置１２１側に送られて記録されることになる。それぞれのプローブによってイベントデータが出力され、記録される。

【００５９】ところで、図２に示したデータ分析比較手段１４２は、本実施例ではクライアント１６１に設けられている。すなわち、クライアント１６１側で発生したイベントデータだけでなく、サーバ１６３側で発生したイベントデータもクライアント１６１側に集中的に集め

られて、性能設計情報１３１との比較が行われる。データ分析比較手段１４２は、測定の開始点から終了点までの経過時間が性能設計情報１３１として記述された条件を満足する場合には“Good”（条件満足）という結果を実際に要した経過時間と共に表示し、条件を満足しなかった場合には“Not Good”（条件不満足）という結果を実際に要した経過時間と共に表示する。

【００６０】図１０は、このプログラム動作検証システムで記録装置あるいは図示しないディスプレイが出力あるいは表示する検証結果の一例を示したものであり、図４に対応させたものである。実際の経過時間としての実行値と設計値および条件を満たしたかどうかの結果とが、クライアント１６１とサーバ１６３の関係で一目でわかるように表示されている。たとえばプログラマあるいは装置の設計者は、サーバ１６３がクライアント１６１から処理の要求を受けてから返答を返すまでの時間が設計値である３秒よりも長い４．２秒かかったことを直ちに知ることができる。すなわち、プログラムコードを実行させたときの“Receive Request FromClient”と“Send Reply To Client”（図４参照）の間の実行性能が性能設計情報１７２で示されている性能条件を満足していないことを知り、必要な対策を採ることができる。

【００６１】第１の変形例

【００６２】図１１は先の実施例の図１に対応するもので被測定計算機の変形例を示したものである。図１１で図１と同一部分には同一の符号を付しており、これらの説明を適宜省略する。先の実施例では被測定計算機１０１と測定装置１０２は別々の装置として構成されていたが、これらを同一の装置で構成することができる。たとえばパーソナルコンピュータとこれに接続された記録装置あるいはディスプレイといったような構成を挙げることができる。この第１の変形例の被測定計算機３０１では、ＣＰＵ３０２内の時計回路３０３を時刻データ１２３の出力用に使用している。また、この時刻データ１２３やＯＳ１１５から出力れるイベントデータ１１８等のデータは測定データ１２４としてメモリ１１２内に格納される。これらのデータが性能設計情報１３１と比較され、その結果が図示しないプリンタでプリントアウトされたり、同じく図示しないディスプレイに視覚的に表示されることになる。

【００６３】この第１の変形例では、被測定計算機３０１を実施例のクライアントと同一のものとして構成することができ、この場合には測定および検証を行うための特別のハードウェアを必要としないという利点が生じる。もちろん、実施例でもその被測定計算機１０１をクライアント１６１と同一のものとして構成することができる。

【００６４】第２の変形例

【００６５】図１２は、本発明の第２の変形例におけるプログラム動作検証システムの構成を示したものである。このシステムはネットワーク４０１によって接続されたクライアント４０２とサーバ４０３によって構成されている。クライアント４０２およびサーバ４０３はそれぞれ図１に示したものと基本的に同一構成の被測定計算機１０１Ｃ、１０１Ｓと測定装置１０２Ｃ、１０２Ｓによって構成されている。そこで、これらの構成部品で図１に示したものと同一のものには同一の数字とアルファベットＣまたはＳを付しており、これらの部品の説明を適宜省略する。測定装置１０２Ｃ内の時計回路１２２Ｃと測定装置１０２Ｓ内の時計回路１２２Ｓとは時刻同期用通信ケーブル４０４で接続されており、両時計回路１２２Ｃ、１２２Ｓの出力する時刻データ１２３Ｃ、１２３Ｓの同期がとられている。

【００６６】クライアント４０２の被測定計算機１０１Ｃ上ではクライアントプログラムが動作し、サーバ４０３の被測定計算機１０１Ｓ上ではサーバプログラムが動作するようになっている。クライアント４０２およびサーバ４０３はそれぞれ単独に測定装置１０２Ｃあるいは１０２Ｓを配置しているので、それぞれ自分の側でイベントデータを時刻データと共に記録し、性能設計情報と比較することができるようになっている。もちろん、ネットワーク４０１で接続先にこれらの比較結果を送ることで、ネットワーク全体のどの装置で処理が滞っているといった情報も取得し、それぞれの計算機で表示したりプリントアウトすることもできる。

【００６７】図１３は、この第２の変形例でクライアントがサーバに所定の処理要求を行って処理結果を得た場合のそれぞれの被測定計算機のソフトウェアイベントとプローブの関係を示したものである。この例は、処理内容的には実施例の図８で説明したクライアント１６１とサーバ１６３の処理と同一であるので、詳細な説明は省略する。先の実施例では、測定装置がシステムに１つだけだったので、図１０で示したような比較結果が１箇所で集中的に行われるが、第２の変形例では自分の装置に関係する分だけを部分的に表示したり、必要によりシステム全体の比較結果を表示することもできる。

【００６８】なお、実施例および第２の変形例では１つのクライアントと１つのサーバの関係を説明したが、１つの計算機から複数の計算機に順次処理が移行していくものや、１つの処理を複数の計算機が分散して行うような場合にも本発明を同様に適用することができる。

【００６９】更に実施例では図１０に示すように、実際の経過時間としての実行値と設計値および条件を満たしたかどうかの結果とがディスプレイに表示されたりプリンタでプリントアウトされるようにしていた。これとは異なり、それぞれの処理に対しての実行値のみがコンポーネントごとに表示されるものであってもよいし、設計値との比較のみが表示されるものであってもよい。

【００７０】また実施例および変形例では測定点間の経過時間を性能設計情報に記述しその結果を表示させる場

合について説明したが、これ以外の性能設計情報、たとえばＣＰＵの稼動率等を併せてこれらの測定結果を合わせて表示するようにしてもよい。

【００７１】

【発明の効果】以上説明したように請求項１および請求項４記載の発明によれば、プログラム動作記述の性能設計情報に記述した内容を基にして測定開始点と測定終了点を抽出するので、たとえばプログラマが性能設計情報として測定の開始点や終了点を判読できるデータを書き込んでおくことにより、被測定プログラムの所望の箇所で測定を開始したり終了させることができる。したがって、プログラムにおける測定点の抽出をソフトウェア技術者あるいは測定実施者が手作業で行う必要がない。また、メッセージの送受信点といったように予め決まったルールを使用したり、ＯＳ自体が用意している機能を利用する場合と異なり、被測定プログラムの所望のコンポーネントの開始点と終了点を任意に指定して測定を行うことができる。そしてこれらの点での処理時刻を求め、これらの差から経過時間を算出して表示するので、実際にかかった時間を被測定プログラムの対応する箇所に対応付けて簡単にチェックすることができる。

【００７２】また請求項２および請求項４記載の発明によれば、プログラム動作記述の性能設計情報に記述した内容を基にして測定開始点と測定終了点を抽出するので、たとえばプログラマが性能設計情報として測定の開始点や終了点を判読できるデータを書き込んでおくことにより、被測定プログラムの所望の箇所で測定を開始したり終了させることができる。したがって、プログラムにおける測定点の抽出をソフトウェア技術者あるいは測定実施者が手作業で行う必要がない。また、メッセージの送受信点といったように予め決まったルールを使用したり、ＯＳ自体が用意している機能を利用する場合と異なり、被測定プログラムの所望のコンポーネントの開始点と終了点を任意に指定して測定を行うことができる。そしてこれらの点での処理時刻を求め、これらの差から経過時間を算出して表示するので、実際にかかった時間を被測定プログラムの対応する箇所に対応付けて簡単にチェックすることができる。更に本発明では、算出した経過時間を設計値と共に被測定プログラムの対応する箇所に表示するので、算出した経過時間が設計値を満足しているか否かを簡単にチェックすることができる。

【００７３】更に請求項３および請求項４記載の発明によれば、プログラム動作記述の性能設計情報に記述した内容を基にして測定開始点と測定終了点を抽出するので、たとえばプログラマが性能設計情報として測定の開始点や終了点を判読できるデータを書き込んでおくことにより、被測定プログラムの所望の箇所で測定を開始したり終了させることができる。したがって、プログラムにおける測定点の抽出をソフトウェア技術者あるいは測定実施者が手作業で行う必要がない。また、メッセージの送受信点といったように予め決まったルールを使用したり、ＯＳ自体が用意している機能を利用する場合と異なり、被測定プログラムの所望のコンポーネントの開始点と終了点を任意に指定して測定を行うことができる。そしてこれらの点での処理時刻を求め、これらの差から経過時間を算出しこれを設計値と比較した判別結果を被測定プログラムの対応する箇所に対応付けて表示するので、チェックするコンポーネントが多いような場合にもそれぞれの所要時間が設計値を満足しているかどうかを短時間にチェックすることができる。

【００７４】また請求項５記載の発明によれば、プログラム動作記述の性能設計情報に記述した内容を基にして測定開始点と測定終了点を抽出するので、たとえばプログラマが性能設計情報として測定の開始点や終了点を判読できるデータを書き込んでおくことにより、被測定プログラムの所望の箇所で測定を開始したり終了させることができる。したがって、プログラムにおける測定点の抽出をソフトウェア技術者あるいは測定実施者が手作業で行う必要がない。また、メッセージの送受信点といったように予め決まったルールを使用したり、ＯＳ自体が用意している機能を利用する場合と異なり、被測定プログラムの所望のコンポーネントの開始点と終了点を任意に指定して測定を行うことができる。そしてこれらの点での時刻を求め、これらの差から経過時間を算出しこれを設計値と比較した判別結果を被測定プログラムの対応する箇所に対応付けて表示するので、チェックするコンポーネントが多いような場合にもそれぞれの経過時間が設計値を満足しているかどうかを短時間にチェックすることができる。

【００７５】更に請求項６記載の発明によれば、請求項５記載の発明による表示内容に加えて経過時間算出手段によって算出した実際の経過時間も表示するようにしているので、この経過時間を設計値と対比することでプログラムの実行について何らかの不具合があったときにその原因を類推することができる。

【００７６】また請求項７記載の発明によれば、１つの被測定プログラムが複数の装置によってコンポーネント単位で実行される場合に、コンポーネントごとの経過時間等の性能指標値を表示したりそれぞれの設計値と比較することが可能になる。

【００７７】更に請求項８記載の発明によれば、１つの被測定プログラムが複数の装置によって実行されるときでも、それぞれの装置におけるコンポーネントごとの検証結果を集中的に調べることができ、プログラムの全体的な動作の把握を短時間に行うことができる。

【００７８】また請求項９記載の発明によれば、１つの被測定プログラムが複数の装置によって実行されるとき、それぞれの装置におけるコンポーネントごとの検証結果を担当する装置ごとに表示するので、これらの装置ごとにハードウェア側の原因を探る等の対処を容易に行

うことができる。

【図面の簡単な説明】

【図１】本発明の一実施例におけるプログラム動作検証システムの構成の概要を示したブロック図である。

【図２】本実施例のプログラム動作検証システムの動作の概要を示した説明図である。

【図３】本実施例のプログラム検証用のクライアントとサーバの関係を示したブロック図である。

【図４】本実施例でプログラマによるプログラム動作記述の例を示した説明図である。

【図５】本実施例でクライアント側のプログラムコードとしてのクライアントプログラムの流れの要部を示した流れ図である。

【図６】本実施例でサーバ側のプログラムコードとしてのサーバプログラムの流れの要部を示した流れ図である。

【図７】本実施例で測定点情報を埋め込んで作成した被測定プログラムを用いたクライアント側の処理を示した説明図である。

【図８】本実施例でイベントを出力するそれぞれのプローブとソフトウェアイベントを対応させて示した説明図である。

【図９】本実施例で被測定プログラムを用いたサーバ側の処理を示した説明図である。

【図１０】本実施例のプログラム動作検証システムで記録装置あるいは図示しないディスプレイが出力あるいは表示する検証結果の一例を示した平面図である。

【図１１】本発明の第１の変形例として被測定計算機の構成を示したブロック図である。

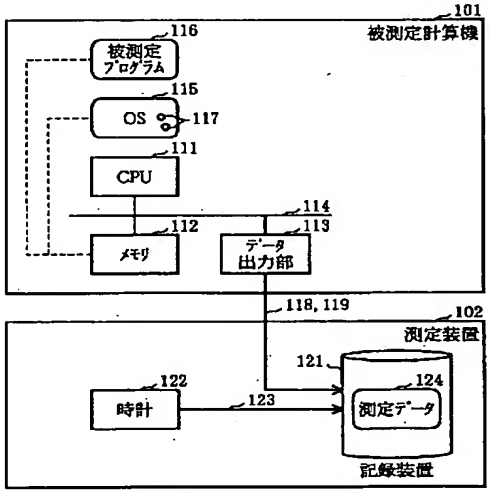
【図１２】本発明の第２の変形例におけるプログラム動作検証システムの構成を示したブロック図である。

【図１３】第２の変形例でクライアントがサーバに所定の処理要求を行って処理結果を得た場合のそれぞれの被測定計算機のソフトウェアイベントとプローブの関係を示す説明図である。

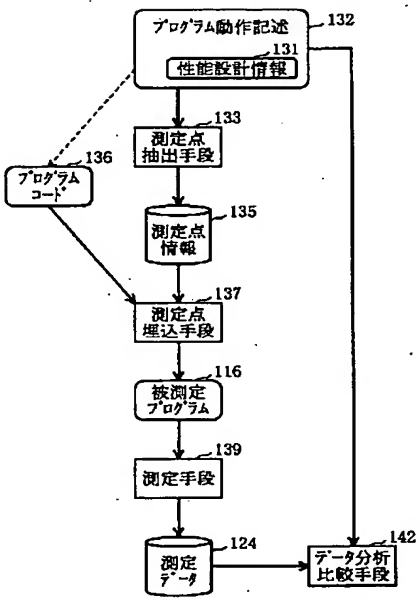
【符号の説明】

１０１、３０１　被測定計算機
１０２　測定装置
１１１　ＣＰＵ（中央処理装置）
１１２　メモリ
１１５　ＯＳ
１１６　被測定プログラム
１１７　カーネルプローブ
１２１　記録装置
１２２、３０３　時計回路
１２４　測定データ
１３２　プログラム動作記述
１３３　測定点抽出手段
１３７　測定点埋込手段
１３９　測定手段
１４２　データ分析比較手段
１６１、４０２　クライアント
１６３、４０３　サーバ
１７１～１７３　性能設計情報
１８１～１８４　測定点
４０１　ネットワーク

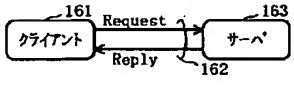
【図１】



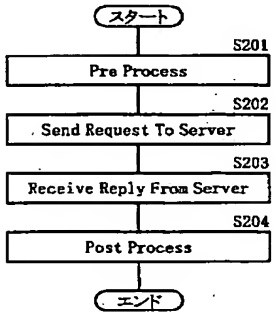
【図２】



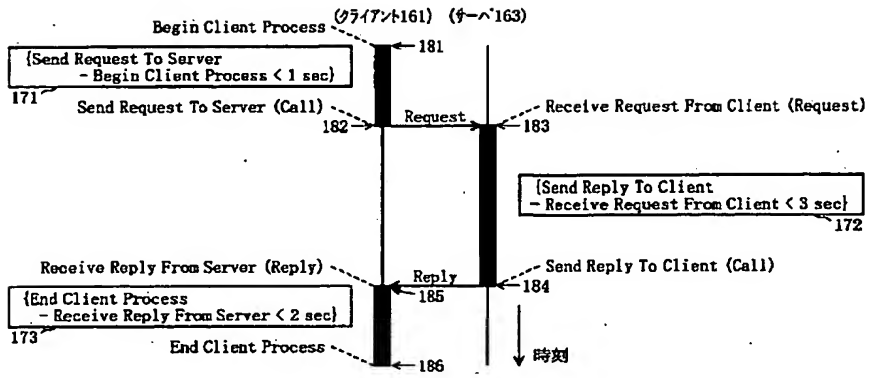
【図３】



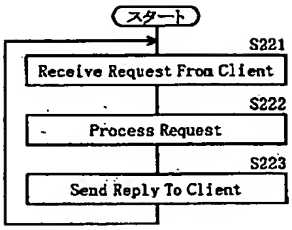
【図５】



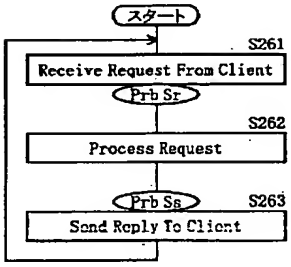
【図４】



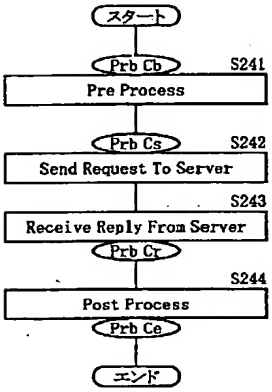
【図６】



【図９】



【図７】

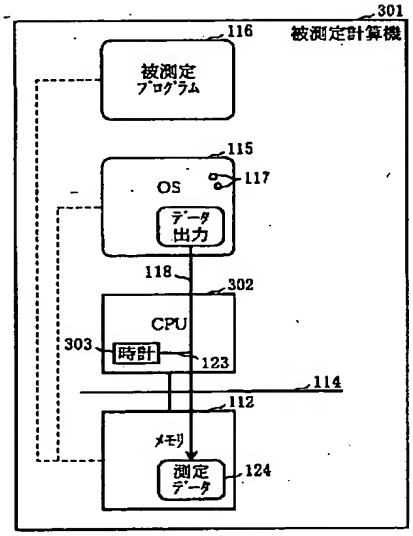


【図８】

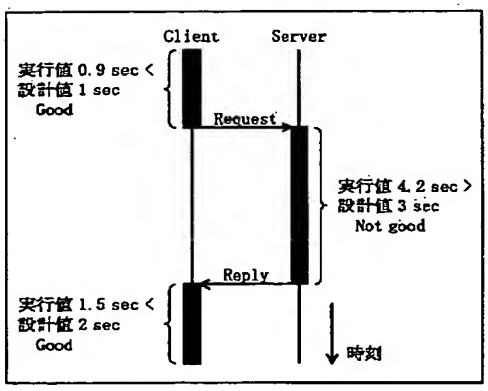
ソフトウェアイベントの説明	イベントを出力するプローブ
...	
クライアント処理開始	Prb Cb
...	
クライアントがサーバに処理要求送信	Prb Cs
クライアントがSleepする	カーネルプローブ
サーバがWake Upされる	カーネルプローブ
サーバがクライアントからの処理要求を受信	Prb Sr
...	
サーバがクライアントに処理結果を送信	Prb Ss
サーバがSleepする	カーネルプローブ
クライアントがWake Upされる	カーネルプローブ
クライアントがサーバからの処理結果を受信	Prb Cr
...	
クライアント処理終了	Prb Ce

時刻

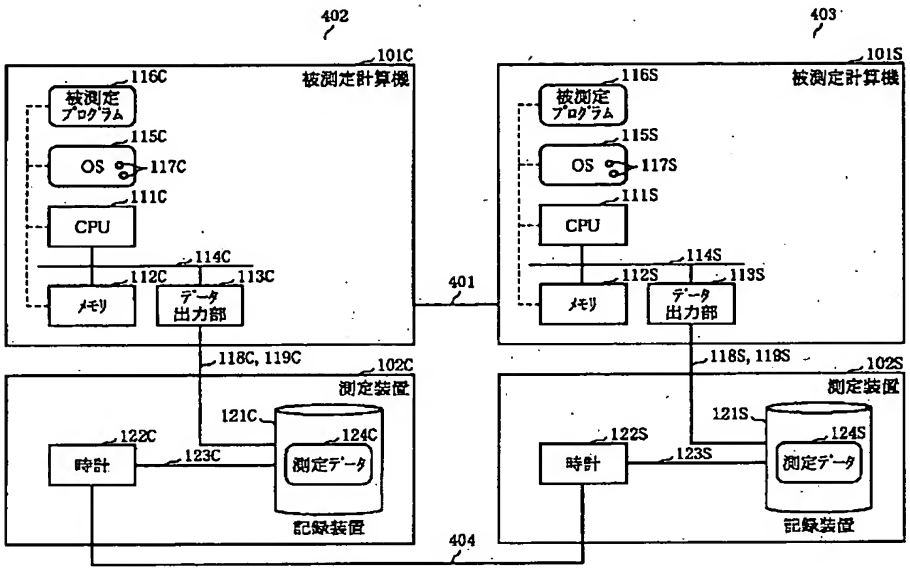
【図１１】



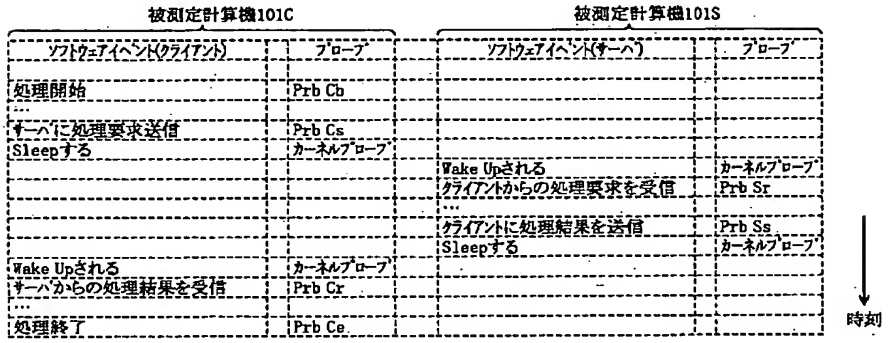
【図１０】



【図１２】


402
403
101C
被測定計算機
116C
被測定プログラム
115C
OS
117C
111C
CPU
114C
112C
113C
メモリ
データ出力部
401
101S
被測定計算機
116S
被測定プログラム
115S
OS
117S
111S
CPU
114S
112S
113S
メモリ
データ出力部
118C, 119C
102C
測定装置
121C
122C
時計
123C
124C
測定データ
記録装置
118S, 119S
102S
測定装置
121S
122S
時計
123S
124S
測定データ
記録装置
404

【図１３】


被測定計算機101C
被測定計算機101S
ソフトウェアイベント(クライアント)
プローブ
処理開始
Prb Cb
サーバに処理要求送信
Prb Cs
Sleepする
カーネルプローブ
Wake Upされる
カーネルプローブ
サーバからの処理結果を受信
Prb Cr
処理終了
Prb Ce
ソフトウェアイベント(サーバ)
プローブ
Wake Upされる
カーネルプローブ
クライアントからの処理要求を受信
Prb Sr
クライアントに処理結果を送信
Prb Ss
Sleepする
カーネルプローブ
時刻